



Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»

Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота

**Розробка розподіленої системи е-комерції з багаторівневим
доступом (на прикладі Aliexpress)**

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту:

№	П.І.Б.	Група	Роль
1	Моторіна Ніколь Павлівна	КН-П-213	Розробник (backend)
2	Волков Данило Андрійович	КН-П-213	Розробник (backend)
3	Поздняков Дмитро Дмитрович	КН-П-211	Розробник (frontend)
4	Червоний Данило Юрійович	КН-П-213	Розробник (frontend)
5	Тимофєєв Євгеній Юрійович	КН-Д-211	Дизайнер UI/UX

Автор звіту: Моторіна Ніколь Павлівна

АНОТАЦІЯ

Дипломний проєкт присвячений розробці Backend частини вебзастосунку — онлайн-гіпермаркету, що є функціональним аналогом платформи Aliexpress. Основна мета проєкту — створення зручного середовища для електронної комерції, яке дозволяє користувачам купувати широкий асортимент товарів онлайн.

У межах проєкту було реалізовано серверну частину системи, яка забезпечує логіку взаємодії між клієнтським інтерфейсом, базою даних та іншими сервісами. Розробка включала створення API для управління товарами, категоріями, кошиком, замовленнями, користувачами, а також системою відгуків, рейтингів та пошуковим механізмом. Забезпечено реалізацію механізмів реєстрації, автентифікації, авторизації та керування сесіями користувачів.

Особлива увага приділялася питанням безпеки: реалізовано шифрування паролів, перевірку прав доступу, токен-автентифікацію (JWT), а також базову систему логування та моніторингу запитів.

У проєкті використовуються сучасні технології:

- **Backend:** ASP.NET Core 8, C#
- **База даних:** MySQL
- **Архітектура:** RESTful API, чиста багаторівнева (підхід "розділення відповідальностей")

Проєкт є актуальним у контексті стрімкого зростання ринку онлайн-продажів. Реалізована система може масштабуватись та адаптуватись до потреб реального бізнесу, має модульну структуру, що дозволяє легко додавати нові функції у майбутньому.

Результатом є стабільна та безпечна серверна частина онлайн-платформи, що взаємодіє з клієнтським інтерфейсом і забезпечує всі необхідні бізнес-процеси електронної комерції.

ЗМІСТ

ВСТУП	4
Технічне завдання до проєкту	5
РОЗДІЛ 1. Загальна характеристика системи	7

РОЗДІЛ 2. Реалізація серверної частини

8

ВИСНОВКИ_____11

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ12

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ13

ВСТУП

Сучасні інтернет-технології активно впроваджуються у сферу електронної комерції, забезпечуючи користувачам зручність у здійсненні покупок, а бізнесу — нові можливості для розвитку. Розвиток онлайн-торгівлі супроводжується зростанням попиту на високоякісні цифрові платформи, які поєднують функціональність, масштабованість та безпеку. Онлайн-гіпермаркети, такі як Aliexpress, виступають яскравими прикладами таких платформ, об'єднуючи великий асортимент товарів із простим та ефективним механізмом покупки. З урахуванням вищезазначених тенденцій, у межах дипломного проєкту було реалізовано вебплатформу — клон онлайн-гіпермаркету Aliexpress. Основна увага приділялась забезпеченню стабільної взаємодії між користувачами та системою, захисту персональних даних, ефективній роботі з базою даних, а також підтримці масштабованості та продуктивності проєкту.

Метою роботи є створення серверної логіки для повнофункціонального онлайн-магазину, що включає модулі авторизації, управління товарами, обробки замовлень, роботи з відгуками, а також інтеграції з клієнтською частиною, реалізованою на Angular. Особливу увагу приділено безпеці, зокрема впровадженню авторизації через JWT, хешуванню паролів.

У вступі описано передумови створення проєкту, його актуальність та основні цілі. У подальших розділах автореферату подано аналіз технічних вимог, описано архітектурне рішення backend-частини системи, обґрунтовано вибір технологій, а також представлено результати тестування та реалізації. У завершенні зроблено висновки щодо ефективності реалізованої моделі та можливостей її масштабування в умовах реального бізнесу.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

Загальний опис

Програмне забезпечення для створення інформаційної системи інтернет крамниці для організації роздрібної інтернет торгівлі.

Призначення

Дипломний проєкт передбачає створення серверної частини онлайн-гіпермаркету — веб-платформи, що забезпечує користувачам доступ до широкого каталогу товарів, можливість їх придбання, а також функції зворотного зв'язку. Основне призначення системи — забезпечити публікацію інформації про доступні товари (оферти продавців), оформлення замовлень (акцепти покупців) через Інтернет, управління контентом онлайн-магазину, збереження даних, управління доступом, а також організацію інтеграції з іншими інформаційними системами за допомогою Web-API.

Вимоги до функціоналу:

- **Інтернет-крамниця:**
 1. Навігація по вітрині магазину
 2. Пошук товарів за різними критеріями
 3. Реєстрація та автентифікація покупців
 4. Формування та управління кошиком
 5. Оформлення замовлень
 6. Можливість залишати питання та відгуки до товарів
- **Управління контентом:**
 1. Перегляд, створення, редагування категорій, товарів та вітрини
 2. Керування користувачами, їх ролями та правами доступу
 3. Модерація коментарів та відгуків
 4. Адміністрування замовлень
- **Реєстрація та авторизація користувачів:**
 1. Підтримка стандартної авторизації через логін та пароль
- **Особистий кабінет покупця:**
 1. Перегляд історії замовлень
 2. Редагування профілю
 3. Керування платіжними та контактними даними
 4. Управління кошиком
- **Взаємодія з іншими додатками (Web-API):**
 1. API для управління акаунтами користувачів
 2. API для товарів, категорій, фільтрації та пошуку
 3. API для формування замовлень і додавання товарів до кошика
 4. API для створення та оцінки відгуків
 5. API для роботи з додатковими елементами вітрини

Архітектура:

У проєкті реалізовано **чисту багат шарову архітектуру** (Clean Architecture), яка забезпечує розділення відповідальностей, високу підтримуваність та масштабованість:

- **Сервер бази даних** — зберігання всіх даних про користувачів, товари, замовлення, відгуки тощо (MySQL).
- **Сервіс Web-API** — RESTful API, реалізований на ASP.NET Core, що забезпечує зв'язок між клієнтською частиною та сервером.
- **Веб-додаток для керування контентом** — окрема адміністративна панель, реалізована через Angular.
- **Веб-крамниця** — клієнтський інтерфейс магазину для покупців, реалізований на Angular та пов'язаний із бекендом через Web-API.

Технології:

- Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології та методології створення:
 - Бази даних – MySQL (реляційна)
 - Сервіс Web-API – REST-API, ASP.NET Core, C#
 - Веб додаток керування контентом – ASP.NET Core Web API, C#
 - Веб додаток для покупця – Angular
- Код реалізовано з дотриманням стандартів якості, включаючи коментування, написання unit-тестів для критичних компонентів та логування запитів. Проєкт розгорнутий локально з можливістю майбутнього перенесення у хмарне середовище.

РОЗДІЛ 1

ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Проєкт реалізує серверну частину онлайн-гіпермаркету у вигляді RESTful Web API, розробленого за допомогою мови програмування C# та фреймворку ASP.NET Core 6. Цей API забезпечує обробку запитів клієнтських додатків, керує бізнес-логікою та взаємодією з базою даних.



Для зберігання інформації використовується реляційна база даних MySQL, вибір якої обумовлений її надійністю, поширеністю та зручністю інтеграції з ASP.NET Core через Entity Framework Core — популярний ORM, що дозволяє ефективно працювати з базою даних через LINQ-запити.

Серверна частина виконує такі функції:

- Реалізація бізнес-логіки онлайн-магазину (управління товарами, замовленнями, користувачами, відгуками тощо);
- Обробка HTTP-запитів і формування JSON-відповідей;
- Забезпечення безпеки через автентифікацію та авторизацію користувачів за допомогою JWT;
- Підтримка масштабованості та надійності системи.

Код серверної частини зберігається у репозиторії за адресою: [репозиторій](#).

РОЗДІЛ 2

РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ

Серверна частина написана з використанням наступних технологій:

- Мова програмування C#
- Фреймворк ASP.NET Core 8
- База даних MySQL
- Середовище розробки Visual Studio
- Технології: EF Core, LINQ
- Додаткові пакети: AutoMapper, Entity Framework Design, Pomelo Entity Framework MySql, BCrypt-Next, Microsoft.AspNetCore.Authentication.JwtBearer, Microsoft.EntityFrameworkCore.Tools

Серверна частина системи реалізована у вигляді RESTful API за допомогою мови програмування C# та фреймворку ASP.NET Core 6. Основою для побудови структури проекту слугувала Чиста Архітектура, що дозволяє відокремити бізнес-логіку від інфраструктурних залежностей, покращити тестованість та забезпечити гнучкість масштабування проекту в майбутньому.

При розробці були дотримані сучасні принципи програмної інженерії, зокрема:

- **SOLID** — для побудови гнучких і підтримуваних модулів;
- **DRY** — щоб уникнути дублювання коду;
- Патерни проектування, такі як Dependency Injection, Mediator, CQRS, які дозволили розділити запити на читання та модифікацію даних, знизити зв'язаність компонентів та спростити обробку складних сценаріїв взаємодії.

Для збереження даних обрано реляційну СУБД MySQL завдяки її надійності та широкій підтримці. Доступ до бази даних реалізовано через ORM Entity Framework Core, що дозволило працювати з даними за допомогою LINQ-запитів та дотримуватись принципів доменно-орієнтованого проектування.

REST API відповідає за:

- обробку HTTP-запитів з клієнтської частини;
- реалізацію бізнес-логіки (робота з товарами, кошиком, замовленнями, відгуками, користувачами);
- обробку авторизації та автентифікації за допомогою JWT;
- взаємодію з базою даних і кешем;
- генерацію структурованих JSON-відповідей.

Усі користувацькі та системні дані, необхідні для роботи додатка, зберігаються у базі даних MySQL. Для роботи з базою даних використовується ORM

(Object-Relational Mapping) Entity Framework Core, що дозволяє зручно виконувати запити за допомогою LINQ. Це значно підвищує швидкість розробки та спрощує читання і підтримку коду.

У проєкті реалізовано повноцінну систему реєстрації, автентифікації та авторизації. Користувач може зареєструватися за допомогою паролю. Паролі зберігаються у зашифрованому вигляді з використанням бібліотеки BCrypt-Next.

```
#region Helper Methods
2 references
private string HashPassword(string password)
{
    return BCrypt.Net.BCrypt.HashPassword(password, BCrypt.Net.BCrypt.GenerateSalt(12));
}

2 references
private bool VerifyPassword(string password, string storedHash)
{
    return BCrypt.Net.BCrypt.Verify(password, storedHash);
}
#endregion
}
```

Також підтримується авторизація з використанням JWT (JSON Web Tokens). Після входу користувачу надається Access Token та Refresh Token, які використовуються для подальшої взаємодії з сервером. Клієнт надсилає Access Token в заголовок кожного запиту до Web API, і якщо він протермінований — оновлює його через Refresh Token.

Підключення JWT реалізовано за допомогою пакету Microsoft.AspNetCore.Authentication.JwtBearer.

```
builder.Services.Configure<JwtSettings>(builder.Configuration.GetSection("JwtSettings"));
builder.Services.AddScoped<IJwtTokenService, JwtTokenService>();

builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    var jwt = builder.Configuration.GetSection("JwtSettings").Get<JwtSettings>();
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = jwt.Issuer,
        ValidAudience = jwt.Audience,
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwt.SecretKey))
    };
});
```

Для мапінгу об'єктів бази даних до DTO використовується бібліотека AutoMapper, яка дозволяє відокремлювати внутрішню структуру БД від даних, що повертаються клієнту.

```
public class OrderItemDto
{
    0 references
    public int Id { get; set; }
    1 reference
    public int OrderID { get; set; }
    0 references
    public int ProductID { get; set; }
    1 reference
    public string ProductName { get; set; } = null!;
    0 references
    public int Quantity { get; set; }
    0 references
    public decimal PricePerItem { get; set; }
    0 references
    public decimal Subtotal { get; set; }
}
```

```
public MappingProfile()
{
    // User mappings
    CreateMap<User, UserDto>();
    CreateMap<UserCreateDto, User>()
        .ForMember(dest => dest.Password_hash, opt => opt.MapFrom(src => src.Password))
        .ForMember(dest => dest.RegistrationDate, opt => opt.MapFrom(_ => DateTime.UtcNow))
        .ForMember(dest => dest.IsActive, opt => opt.MapFrom(_ => true))
        .ForMember(dest => dest.Rating, opt => opt.MapFrom(_ => 0.0));
    CreateMap<UserUpdateDto, User>()
        .ForAllMembers(opts => opts.Condition((src, dest, srcMember) => srcMember != null));

    // Product mappings
    CreateMap<Product, ProductDto>()
        .ForMember(dest => dest.CategoryName, opt => opt.MapFrom(src => src.Category.Name))
        .ForMember(dest => dest.SellerName, opt => opt.MapFrom(src => src.Seller.Name));
}
```

Валідація вхідних даних здійснюється перед збереженням до бази. Це дозволяє мінімізувати помилки ще до моменту збереження. Для цього використовується класична перевірка атрибутів або додаткові кастомні перевірки.

Код є асинхронним там, де це доцільно, наприклад, при зверненнях до бази даних або зовнішніх сервісів.

Пошук товарів реалізовано з можливістю фільтрації та сортування, при цьому структура коду дозволяє легко розширювати критерії пошуку, додаючи мінімальні зміни.

ВИСНОВКИ

У межах дипломного проєкту було реалізовано єдиний Web API-сервіс, що обслуговує як клієнтську частину (вітрину магазину), так і адміністративний інтерфейс. Серверна частина побудована відповідно до принципів Чистої архітектури, яка забезпечує чітке розділення відповідальностей, слабе зв'язування між компонентами, легкість супроводу та можливість масштабування.

Оснoву технологічного стеку становлять C#, ASP.NET Core 8, Entity Framework Core, а база даних MySQL розгорнута на AWS RDS, що забезпечило високу продуктивність, надійність зберігання даних і зручність адміністрування.

У проєкті використано патерни Unit of Work та Generic Repository, що дозволяють централізовано управляти транзакціями та ефективно працювати з даними. Сервіси (наприклад, UserService, ProductService) інкапсулюють бізнес-логіку, залишаючи контролери «тонкими» і чистими.

Безпека системи реалізована через JWT-аутентифікацію, хешування паролів з використанням BCrypt, а також підтримку авторизації через сторонні сервіси, такі як Google. Для управління обліковими записами застосовується ASP.NET Identity, що дозволяє керувати ролями, автентифікацією та збереженням облікових даних.

Для зручного мапінгу моделей використовувався AutoMapper, який дозволив ізолювати DTO від внутрішньої структури бази даних, спростивши підготовку даних для клієнтської частини.

Важливою складовою проєкту є асинхронність запитів, яка дозволила покращити продуктивність при роботі з базою та зовнішніми сервісами.

Додатково реалізовано валидацію вхідних даних з використанням атрибутів і кастомних правил, що забезпечує захист API на рівні моделі.

Також було впроваджено базову систему логування, яка дозволяє відстежувати запити, помилки та інші важливі події в життєвому циклі застосунку.

Під час розробки дотримувались сучасних принципів програмування: SOLID, DRY, залежності впроваджувались через Dependency Injection. Завдяки цьому вдалося створити стабільний, гнучкий і масштабований серверний застосунок, готовий до використання в реальному бізнес-середовищі.

Проєкт має великий потенціал для розвитку — можлива інтеграція з платіжними системами, реалізація повноцінної адміністративної аналітики, розширення API, а також автоматизація CI/CD.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

- [C#](#)
- [.NET Core](#)
- [ASP.NET Core](#)
- [Entity Framework Core](#)
- [AutoMapper](#)
- [Visual Studio](#)
- [MySQL](#)
- [Postman](#)
- [BCrypt-Next](#)
- [JwtBearer](#)

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [Microsoft technical documentation](#)
- [Stackoverflow](#)
- [ChatGPT](#)
- [YouTube](#)
- [EF Core Tutorial](#)
- [Google authentication documentation](#)
- [GitHub](#)
- [Metanit](#)